Reconfigurable High Performance Architectures: How much are they ready for safety-critical applications?

D. Sabena, L. Sterpone
Dipartimento di Automatica e Informatica - DAUIN
Politecnico di Torino
Torino, Italy

Mario Schölzel, Tobias Koal, H.T. Vierhaus
Brandenburg University of Technology
Department of computer science
Cottbus, Germany

S. Wong
Computer Engineering Laboratory
Delft University of Technology
Delft, The Netherlands

R. Glein, F. Rittner, C. Stender
RF and Microwave Design Department,
Fraunhofer Institute for Integrated Circuits (IIS)
Erlangen, Germany

M. Porrmann, J. Hagemeyer
University of Bielefeld
Bielefeld, Germany

***Abstract*—Reconfigurable architectures are increasingly employed in a large range of embedded applications, mainly due to their ability to provide high performance and high flexibility, combined with the possibility to be tuned according to the specific task they address. Reconfigurable systems are today used in several application areas, and are also suitable for systems employed in safety-critical environments. The actual development trend in this area is focused on the usage of the reconfigurable features to improve the fault tolerance and the self-test and the self-repair capabilities of the considered systems. The state-of-the-art of the reconfigurable systems is today represented by Very Long Instruction Word (VLIW) processors and reconfigurable systems based on partially reconfigurable SRAM-based FPGAs. In this paper, we present an overview and accurate analysis of these two type of reconfigurable systems. The content of the paper is focused on analyzing design features, fail-safe and reconfigurable features oriented to self-adaptive mitigation and redundancy approaches applied during the design phase. Experimental results reporting a clear status of the test data and fault tolerance robustness are detailed and commented.**

I. Introduction

Reconfigurable systems and processors are increasingly used in several application scenarios. The most important characteristic lies in the fact that they can be easily tuned to match the specific requirements of the target application, e.g., in terms of power consumption, size, and performance; given these features, is some cases reconfigurable systems and processors are more convenient than traditional processors [1].

Reconfigurable systems are acquiring an increasing interest in the domain of safety-critical applications, for example in space and avionic applications, due to the capability of reconfiguring the system during run-time execution; moreover the high computational power of modern Field Programmable Gate Arrays (FPGAs) makes these devices suitable for data processing. Finally, reconfigurable systems must also guarantee the abilities of self-awareness, self-diagnosis and self-repair in order to face with errors due to the harsh conditions typically existing in some environments [2].

In this paper we present an overview and an accurate analysis of the two major type of reconfigurable systems: Very Long Instruction Word (VLIW) processor and Dynamic and Partial reconfigurable Platforms (DRPMs). More in details, we propose a summary of the last developed work in this area, with particular emphasis on the motivations that have driven the developers, and on the application areas of the proposed methods.

II. Introduction To On-board Processors

An example of reconfigurable system based on modern SRAM-based FPGAs is the Fraunhofer On-Board Processor consisting of two space grade Xilinx Virtex-5QV FPGAs, which presents an overview of the dependable reconfiguration system. This platform does not require any additional programmable device like a microcontroller or an anti-fuse FPGA for the reconfiguration of the Virtex-5QV. An initial-configuration, stored in a non-volatile Magnetoresistive RAM, ensures the fail-safe reconfiguration of the FPGAs. Each FPGA is able to reconfigure itself due to partial reconfiguration and can reconfigure the other one completely. The initial-configuration consists of a static part and a dynamically reconfigurable part. The static part includes a System on Chip (SoC). The main task of this SoC is the reconfiguration of both FPGAs, the fault management and the virtual Telemetry/Telecommand processing. The System on Chip starts after the initial-configuration of both FPGAs at power up. If the startup procedure completes successfully one

978-1-4799-3415-7/14/$31.00 ©2014 IEEE

of the two FPGAs can be reconfigured entirely for our high-speed digital signal processing experiments. For a fault prediction we performed a radiation analysis by calculating the upset rates of the FPGAs primitives at different solar conditions. For a fault prediction we performed a radiation analysis. We calculated the upset rates of the FPGAs primitives at different solar conditions. With our proposed internal Block RAM radiation particle sensor we mitigate Single Event Upsets in the FPGA. During a harsh solar condition due to solar flares the Block RAM sensor triggers our self-adaptive mitigation and a mitigation scheme (e. g. Triple Modular Redundancy) is applied automatically.

The Fraunhofer On-Board Processor (FOBP) is part of the scientific payload of the Heinrich Hertz communication satellite. This satellite is planned to be launched into a Geostationary Earth Orbit (GEO) in 2018 followed by an operation time of 15 years. During the in-orbit verification the Fraunhofer IIS will perform various communication experiments with the FOBP.

Figure 1 depicts the four main hardware modules of the FOBP: A Radio frequency card (analog front-end receiver, analog front-end transmitter and clock distribution), a power supply unit with high voltage High Power Command (HPS) and Bi-level Switch Monitor (BSM) and two Digital Signal Processing (DSP) cards. The intermediate center frequency (IF) of the analog down-converted uplink inputs and outputs before up-conversion is 1530°MHz. The bandwidth of input 1 (I1) and output 1 (O1) is 36°MHz. Input 2 (I2) and output 2 (O2) for high data rate transmissions carry signal with a bandwidth of 450°MHz. A 1MHz carrier for a virtual Telemetry/Telecommand (vTM/TC) is mixed in the signals of both paths. We control and monitor the whole FOBP via this vTM/TC link, which is in-band of the user data link. For these experiments we use the Virtex-5QV space-grade FPGAs with the following benefits: They are robust against Total Ionizing Dose (TID) greater than 1°Mrad (Si), immune against destructive Single Event Effects (SEEs) and radiation-hardened against temporary SEEs.

III. Introduction To VLIW processors

Very Long Instruction Word (VLIW) processors are adopted in a large range of embedded signal processing applications, mainly due to their capability to achieve high performances with low power consumption and clock frequency. The robustness of VLIW processors is extremely important, at the same time there is an increasing demand for efficient and optimal test techniques able to detect permanent faults in VLIW processors [1][3]. These algorithms would be effective in identify erroneous resources and to perform an on-site reconfiguration and correction of the faulty module. Software-Based Self-Test (SBST) methods are an efficient approach to individuate faults into a processor both at the end of the manufacturing phase or during the life of the circuit.

In processor design, we can distinguish between two extremes between "high performance and low flexibility" or "low performance and high flexibility", namely application-specific processors or general-purpose processors, respectively. Reconfigurable processors are envisioned to provide the best of both world by achieving high performance with high flexibility, i.e., the support of many different types of operations. The MOLEN processor [4] represents an approach in which arbitrary accelerators can be tied to any general-purpose processor via a one-time ISA extension. The accelerators provide the high performance and the support for any accelerator provides the flexibility. However, the design of accelerators requires lengthy design times (manual) in order to achieve the "best" performance and "lowest" energy consumption. On the other hand, automatic tools provide a way to quickly generate accelerators, but they are usually hampered by requiring rewriting of the original application code for them to work efficiently. In this light, the authors in [5] proposed an intermediate approach to utilize a parameterized VLIW softcore, called ρ-VEX, that allows the use of existing compilers to quickly generate code and exploit the ILP (instruction- level parallelism) inherent in kernel code using the VLIW to achieve "high-enough" performance. This is similar to utilizing horizontal microcode [6]. The parameterization allows for quick adaptation to different applications and thereby achieving the needed flexibility. Additional benefits in utilizing a VLIW stem from the fact that the instruction scheduling is performed in the compiler removing the need for a complex and power-hungry instruction scheduler as in superscalar RISC processors. The initial project presented in [4] quickly morphed to the design of a stand-alone ρ-VEX softcore that is envisioned to support dynamic, i.e., run-time, reconfiguration of its parameters. The main parameter that is being exploited is ILP. In [7], a design was proposed that allowed 8 VLIW datapaths to be constructed as a single 8-way, 2 times a 4-way, 4 times a 2-way VLIW processor(s) or any combination. Moreover, unused resources can be clock gated in order to reduce power consumption. This allowed for a dynamic exchange between ILP and TLP (thread-level parallelism) meaning that a single design can be seen as a single core or as multiple many-/multi-cores. This is a departure from the current modus operandi in using compilers to optimize code to run on different microarchitectures to a scenario where the core can dynamically adapt itself to the requirements of the application(s) or the environment it is being used. At the same time, the reconfigurability allows for the ρ-VEX to overcome the issues of past VLIWs that prevented them from becoming mainstream, such as unbalanced datapaths and a high number of NOPs.

A. Generic Binary

Having the ability to exchange between ILP and TLP does not yet make the proposed ρ-VEX core dynamic. In the dynamic scenario, one would like to be able to switch the execution of an application transparently between 2-, 4-, and 8-way modes. Without introducing any support, one has to compile three versions of the application code and switch between them when necessary. However, this approach has several caveats. First, the code cannot be interrupted as one cannot guarantee that the interrupted program point is exactly the same as in another version of the application code. Second,

depending on the frequency of switching between the modes (and assuming correct check-pointing was implemented), the loading of codes can greatly hamper performance. Therefore, [8] introduced the notion of generic binaries that is a single binary that can be executed on multiple-way VLIW processors. This allowed for the support of interrupts and removed the need for program loading when the number of ways need to be changed. This flexibility does not come for free as a reported performance hit between 10% and 30% had to be paid. However, the authors explained that this was due to the closed-source nature of their utilized compiler as the necessary improvement to reduce this performance hit could not be implemented.

B. Fault Tolerance

Fault-tolerance support in the ρ-VEX processor can be implemented in many ways. One approach is to incorporate hardware support for SEU (single event upset) errors that was presented in [9]. More specifically, the presented approach allowed the fault-tolerance circuitry to be dynamically turned on or off depending on the need for protection of the code. This means that when protection is not needed, power consumption can be reduced. In this way, the average power consumption can be reduced without any consequence of degrading the fault-tolerance coverage. The paper also demonstrated that the area overhead and cycle time degradation, compared to an always-on fault-tolerance support, is minimal.

With the notion of generic binaries, a dynamic software approach can also be utilized to provide protection to critical code sections within an application using the dynamic VLIW core introduced in [4]. For example, critical code sections can be duplicated or triplicated depending on the level of protection that is needed to be executed on the lower number of ways-VLIW cores. In this way, performance can be dynamically exchanged for (code) protection. Another example pertains the dynamic lowering of the number of ways of the hardware in executing an application to allow for checker threads to run on the "freed" datapaths. When the other datapaths need to be checked, the application code can be dynamically moved to the other datapaths. In this manner, the application can be kept running and be still responsive. This is different from current approaches that require an actual context switch to allow for checker threads to execute on the core. This introduces a lot of overhead that reduces performance and responsiveness of the application. Finally, we would like to note that the pro- posed dynamic hardware fault-tolerance support is orthogonal to the proposed software approaches. This means that they can be used at the same time in order to provide the necessary level of fault- tolerance.

C. ρ-VEX v3.0

After two versions of the ρ-VEX softcore (both available for download and free to use for academic purposes) the ρ-VEX v3.0 is nearly ready for release. While the earlier releases required resynthesis in order to run different applications, v3.0 is intended to support the dynamic loading of applications. It is a SoC built on top of the GRLIB platform [10]. It can address DDR memory attached to the FPGA through separate reconfigurable data and instruction caches and supports any peripheral that can be connected to the AMBA bus. This includes the programmable timer and interrupt controller that can be found in GRLIB. The core has been extended with a vectored trap controller unit that connects to GRLIB's interrupt controller. This way, the platform supports exception handling and interrupt-driven software. Additionally, we have designed a debug unit for the ρ-VEX core. Using the AMBA-JTAG interface found in GRLIB, we can connect to it from a host PC and debug programs running on the platform. To this end, we have developed the ρ-VEX target for the GDB (GNU DeBugger) and modified an open source JTAG - RSP (Remote Serial Protocol) bridge program to connect to our debug unit. In order to provide run-time support for real-world applications, this design is also intended to run μc-Linux (as virtual memory is not supported) and progress is being made in porting the kernel and standard C library from that distribution to ρ-VEX.

IV. Dependability of Partially Reconfigurable Hardware

Dependability in computer-based systems can be defined in attributes, threats, and means to attain dependability. The main attributes of the dependability are: Availability, reliability, safety, integrity, and maintainability. A Concept for threats of losing the service is the Fault-Error-Failure chain. Means to attain dependability can be classified in: Fault prevention, fault tolerance, fault removal and fault prediction. In this section, we discuss structures and methods to achieve a dependable reconfiguration system [11][12].

A. Initial Configuration and Start-Up

The digital signal processing of the FOBP includes two FPGAs without additional configuration logic devices. In order to attain a high reconfiguration reliability and to avoid single points of failure, we propose an initial-configuration framework, which enables partial self-reconfiguration as well as a complete reconfiguration of the FPGAs. This framework empowers fail-safe reconfiguration and reduces the dependency on external configuration logic. These advantages arise out of the coexistence of the initial-configuration with a state-of-the-art configuration method (FPGA1 configures FPGA2 and vice versa). The initial-configuration needs only one additional protected non-volatile memory per FPGA, e.g., Magnetoresistive RAM (MRAM). See [13] for reliability analysis and implementation details. At FOBP power on, both FPGAs are configured with identical bit files stored in the MRAMs. Following both FPGAs negotiate a master, which is responsible for reconfiguration, fault management, and vTM/TC processing. Fig. 1 depicts the configuration of the FPGAs after initial configuration and start up. In this case, FPGA1 was negotiated as master and is still configured with the initial-configuration. FPGA2 can be reconfigured completely with any firmware send via the vTM/TC link. This configuration concept allows an effective and reliable maintenance in terms of software and firmware updates.

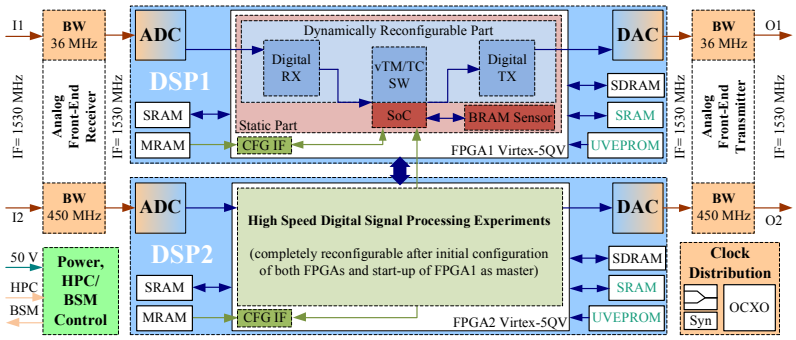

Figure 1. The scheme of the FOPB Reconfigurable and Fault Tolerant System

B. System on Chip and Tasks

The System on Chip (SoC) is located in the static part of the initial-configuration and is based on a 32-bit microprocessor. Most likely, we will use the LEON3FT from Aeroflex Gaisler. The hardware and the boot code of the SoC is part of the static area in the initial-configuration and is therefore unchangeable. Nevertheless, it is possible to update the application software (e.g., the vTM/TC processing code). The three main tasks of the SoC can be described as follows.

The first main task is the reconfiguration of both FPGAs via the Internal Configuration Access Port (ICAP) of the own FPGA and the Select MAP (SMAP) interface of the FPGA on the other DSP card. Additionally, the reconfiguration task includes scrubbing with read back for both FPGAs. Fault management is the second main task and is realized with a Fault Detection, Isolation and Recovery (FDIR) system. One part of the FDIR system is a fault detector, which is capable of detecting certain faults in several components inside and outside of the FPGA. A detected fault is reported to a fault management unit, which calculates countermeasures. A countermeasure can be a reset of a certain firmware or hardware unit, (partial) reconfiguration of the FPGAs or switching into safe mode. For the design of the FDIR system we took the radiation effects of the FPGA into account, see Section III-C.

The implemented fault management increases the availability and reliability of the FOBP. Telemetry and telecommand of the FOBP is the third main task. Commands and data (e.g., a bit file or software update) are sent from the ground station to the FOBP via the vTM/TC uplink. The state of the FOBP and response messages are sent back to earth via the vTM/TC downlink periodically.

C. Radiation Analysis

Temporary SEEs are the dominant fault effects of radiation-hardened SRAM-based FPGAs in orbit. The focus of this section are Single Event Upsets (SEU) as part of SEEs. SEUs occur in every FPGA primitive. We calculated the upset rates of the different primitives: Configuration controller, SRAM configuration memory, flip flops, multiplier and Block RAM (BRAM). Based on the radiation conditions in GEO and an aluminum shielding of 7°mm we used the widely accepted tool Cosmic Ray Effects on Micro-Electronics (CREME96) to calculate the upset rates for five different solar conditions (Solar Minimum, Solar Maximum, Worst Week, Worst Day and Peak 5 Minutes) [14].

One result of the upset rate calculation is the high sensitivity of the non-hardened BRAM regarding protons and heavy ions. We take advantage of this special property of the BRAM to perform a run-time fault prediction in orbit. Since environmental changes have an order of magnitude impact on upset rates, it is beneficial to adapt the redundancy to the current environment rather than design for the worst-case expected environment. Since BRAM upset rates are higher than other upset rates, we can use BRAM upsets as a sensor to characterize the environment. We perform a self-adaptive SEU mitigation based on this environment characterization. In a case study, we show that it is possible to triplicate the data throughput at the Solar Maximum condition (no flares) compared to a Triple Modular Redundancy implementation of a single module. We also show the decreasing Probability of Failures Per Hour by 2 x 10^4 at flare-enhanced conditions compared with a non-redundant system. The proposed BRAM radiation sensors can be utilized in a user design without the need of additional BRAMs. This is possible since the sensor reads the Error-Correcting Code (ECC) data of the accessed word and processes it [15].

V. DRPM for Hardware Satellite Payload

Safety critical missions, driven by space and avionic applications, are increasingly attracting the usage of reconfigurable systems due to low non-recurring engineering costs, reconfigurability and large number of logic resources they provide. Among the various reconfigurable systems developed, the ones implemented on SRAM-based FPGAs are the most effective to cope with the demanding on-board processing capabilities. SRAM-based FPGAs are characterized by large gate counts and they provide a flexible platform to implement a complete System-on-Chip (SoC) on a single device. Besides, SRAM-based FPGAs are suitable for applying dynamic reconfiguration, thus enabling the FPGA to be partitioned into static region (SR) and partially reconfigurable (PR) region, where the static region contains the components that are not changed during run-time, such as interface or memory controllers. Vice versa, the partially reconfigurable

region offers resources for various different hardware modules that can be loaded at run-time according to the needs of the target application. These dynamic components are represented by so called dynamic partially reconfigurable modules (PRM). At run-time one or several instances of a PRM can be placed in the PR region, while the communication between the PR region and the SR region is realized by a communication macro, which is explicitly designed to cope with the special needs of dynamic reconfiguration [16]. For the purpose of this work, we implemented a tiled PR region according to the scheme illustrated in Figure 2. In this scenario, the PR region is divided into reconfigurable tiles, which define the atomic unit of partial reconfiguration. Each instance of a PR module is composed by a set of contiguously aligned tiles, where each tile consists of a set of static and reconfigurable resources. The static resources are available for the instantiation of PR modules at run-time.

As documented by several research works on the field, one of the most relevant problems in adopting SRAM-based FPGAs in radiation-harsh environment is the dangerous effects induced by radiation particles such as atmospheric neutrons and heavy ions [17]. These particles may induce non-destructive loss of information within the system provoking Single Event Upsets (SEUs) phenomena that may affect the functionality of the implemented system. In a SRAM-based FPGA, SEUs may affect both the memory elements used by the design the FPGA implements as well as the FPGA's configuration memory. Several works demonstrated the sensitivity of SRAM-based FPGAs to the SEUs induced by high energy particles [18][19] besides various hardening techniques have been developed to specifically address the mitigation of such effects when circuits are implemented on these devices. In this paper we show the results of a neutron testing radiation experiment on a DRPM system which has been evaluated in two different versions: unhardened, thus implemented using commercially available tools, and hardened according our developed hardening design flow based on suitable place and route implementation of the system on the SRAM-based FPGA device.

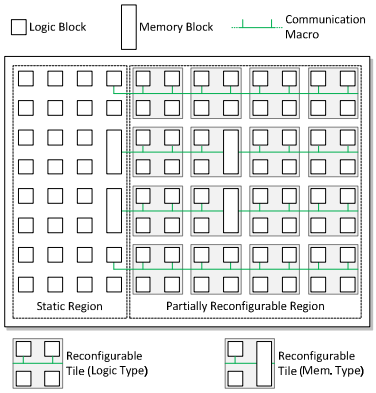

Figure 2. The Dynamically Reconfigurable Processing Module (DRPM) main architecture overview.

The DRPM system has been implemented on Xilinx SRAM-based FPGAs: a Xilinx Virtex-4 FX100. The resources have been placed according to the scheme reported in Figure 2. Radiation testing were performed in the VESUVIO neutron facility at ISIS, Rutherford Appleton Laboratories (RAL) in Didcot, UK. The experimental results we gathered show an evident reduction of the Single Event Functional Interrupts (SEFIs) effect observed on the hardened version of the DRPM with respect to the original unhardened version.

VI. Self-Repair of Reconfigurable Processors

Nano-scaled hardware provides high performance at low power consumption. Unfortunately, it becomes also more susceptible to aging effects causing permanent faults [20]. This increases for long living embedded systems the probability to fail during their mission time. Fault-tolerance techniques targeting permanent faults will improve the reliability of such systems. Handling of permanent faults is usually accomplished by hardware redundancy. While passive hardware redundancy techniques are used for fault masking, active hardware redundancy allows for a reconfiguration of the system, such that faulty components are taken out of operation [21]. This allows for handling of multiple permanent faults at reasonable costs, because a faulty component that is taken out of operation by a reconfiguration will not further contribute to the results of the system.

In superscalar processors, redundant hardware is inherently available. Usually it is used for achieving high performance by parallel execution. But, obviously, it can be also used for fault tolerance purposes. Thereby, the inherently available hardware redundancy can be administrated at various levels of the system stack and at different granularity levels as it is depicted in Figure 3.

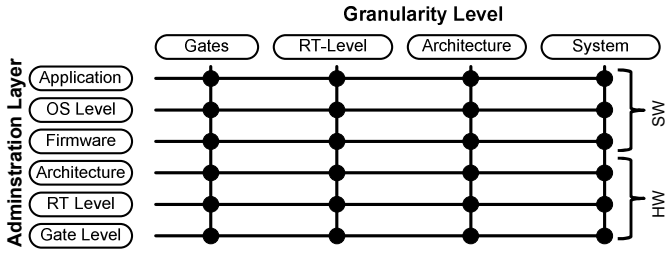

Figure 3. Classification of hardware-redundancy according to the used granularity of the components and the system layer for administration.

The granularity level specifies the size of redundant components that can replace each other. The size of these components ranges from processor cores down to simple gates. The administration of these redundant components can take place at various levels of the system stack. Thereby, the administration can be divided roughly into software-based (SW) and hardware-based (HW) administration schemes. As an example consider dynamically scheduled superscalar processor architectures where the hardware scheduler is extended to duplicate operations that are executed by different functional units. There, the redundantly available functional units are administrated in hardware. The same duplication of operations can be achieved in software by a compiler that generates a program for a very long instruction word (VLIW) processor. Hardware-based administration at low granularity levels requires a strong administrative overhead of more than 100% only for switches and control of redundant hardware. This administrative overhead is reduced significantly, if the granularity is coarsened. However, coarser-grained granularity

also means that by a single fault a huge functioning portion of the system is taken out of operation. The advantage of software-based administration is that the granularity-level can be reduced substantially without significantly increasing the hardware overhead. The basic idea of the software-based reconfiguration is to reconfigure the application in the program memory, such that faulty components in the data path are no longer used. Such a software-based administration can be applied only to statically scheduled processors, e.g., VLIW processors, where the program has control over the used hardware components. Some early work on a software-based reconfiguration of a programmable application specific data path was published by Guerra [22] and Karri [23]. There, various schedules of the same program were generated in advance for different fault states of the data path. If a particular permanent fault occurs in the field, then the corresponding schedule that avoids the usage of the defective component was selected for execution. This solution becomes impractical for larger applications. A few approaches were recently proposed where the adaption of the software is done autonomously in the field. In [24], Meixner has proposed the usage of a compiler in a multicore system with a general-purpose processor and several simple cores. The general purpose core is used for recompiling the user applications for the simple cores in the field. Various detours were presented for this purpose, for example for avoiding the usage of particular registers, operators, and bypasses in the simple cores. However, the required memory overhead for the used gcc-compiler, source code, and libraries is not reported. The complexity of the compiler-backend can be reduced significantly by adapting the binary code of the user application of a VLIW processor. Various reconfiguration techniques for VLIW processors programs were presented. For simple adaptation schemes the repair routine (i.e., the compiler-backend) is implemented with less than 300 VLIW instructions and executes within less than a second for adapting some hundred thousands of instructions. Moreover, it was shown that with different versions of the repair routine (each version uses different resources) a single VLIW core can perform the reconfiguration of its own program code. The granularity of the software-based reconfiguration was lowered down to simple 2:1-multiplexer structures for read ports and bypasses without introducing any additional hardware overhead. By this finer granularity the performance degradation of the processor could be reduced significantly. Instead of about 300% of run time overhead, only 20% runtime overhead were achieved.

The fine-grained software-based reconfiguration of the processor requires an error detection and diagnosis mechanism that can be carried out in the field at that granularity-level that is used by the reconfiguration techniques presented. Error detection may be done on-line concurrently with the execution of the user application for detecting temporary and permanent faults or periodically (off-line) for detecting permanent faults only. Usually periodically testing can be performed at lower granularity with less overhead than concurrent error detection. Two major strategies may be used for this purpose, which were originally developed for manufacturing test and diagnosis: built-in self-test (BIST) and software-based self-test (SBST). Diagnostic BIST approaches usually collect test responses as signatures on-chip. But diagnosis of them is done off-chip, because determining the fault site based on the signature is a computational complex and time consuming task. The runtime of the algorithms used for such a diagnosis is usually in the range of minutes, measured on workstations. Moreover, diagnostic information is provided at structural gate level, but software-based reconfiguration schemes need coarser grained fault state information at functional level. For example, does a particular adder work properly, or can a particular register be used. For coarser grained BIST approaches the diagnostic complexity is reduced, such that it can be on-chip. For example, in [25] test patterns for particular components of a processor are stored in an on-chip ROM, such that they can be tested during idle cycles. Vierhaus et al. proposed a scan-based BIST technique for VLIW processors that employs the redundancy in the data path at slot level. Test patterns are generated with LFSR structures, but test responses are not compared with pre-computed test responses. Rather, the same test pattern is applied to all slots of a VLIW processor, and the obtained results are compared with each other. Similar ideas have been developed. The test patterns are stored in the program memory as if they were instructions. Due to the used VLIW processor architecture, these test patterns can be fetched directly into the pipeline for testing the data path as if they were normal instructions. Test responses are checked at the end of the pipeline by comparing them with each other. The disadvantage of these approaches based on a majority vote is that they fail for multiple faults in the slots of a VLIW processor.

Basically, a SBST is well suited for providing functional fault state information in the field. Test programs for various processor components were recently developed for read ports, register files, TLBs, and branch prediction units. By putting together all of these test programs, high fault coverage for complex processors may be achieved. Unfortunately, a good diagnostic test program is not obtained easily in this way.

The diagnostic capability of such a test program is limited, because even a simple test program uses many processor components, such that a faulty component is not uniquely identified. Diagnosis a diagnostic test routines composed of many test programs can be also effective. Each test programs utilizes only few components of the processor, such that a failing test program refers to a small set of faults only. On the other hand, all the test programs together should cover as many faults as possible. In order to achieve both goals. From the pass/fail information of each test program, the set of fault candidates is isolated by using fault trees. A major problem of this approach is the generation of small diagnostic test programs. A method for automatically improving diagnostic test programs by an iterative improvement phase using genetic algorithms can be applied [26]. Unfortunately, both diagnostic SBST approaches provide diagnostic information at structural gate level, which is not the desired information for software-based reconfiguration techniques. Moreover, the SBST programs are used as static programs. This becomes a problem, when multiple faults must be localized correctly in a processor. For example, suppose the test program for an adder uses a

particular register for loading the operand values. If this register becomes faulty, then the adder cannot be tested anymore, because there is no underlying hardware-based reconfiguration scheme that changes the configuration in a transparent way for the software. For this purpose diagnostic and adaptive SBST routine is proposed for a simple VLIW processor, where software-based reconfiguration techniques are used for adapting also the test program to the current fault state of the processor. By this it is ensured that the diagnostic test routine can determine in a reliable manner the faulty component, even if there are already other faults present in the processor. The diagnostic test of most components of the VLIW-processor is accomplished within a few milliseconds and a total test routine size of about 26 KByte. Unfortunately the adaptation of the test programs must be done by an external administration processor, which requires an additional hardware overhead of 5%. Hence, diagnostic software-based self-test and reconfiguration of VLIW processors for handling permanent faults can be accomplished with low hardware-overhead, moderate program memory overhead and within a short execution break, whereby the software-based reconfiguration time dominates by far the software-based self-test time.

VII. SOFT ERRORS ON DRPM

Radiation experiments were performed in the VESUVIO neutron facility at ISIS. We irradiated the devices with the available spectrum that has already been demonstrated to be suitable for emulating the atmospheric neutron flux. The available flux was about 5.89·105 n/(cm2·s) for energies above 10MeV. Irradiation was performed at room temperature with normal angle of incidence. The beam was focused on a spot with a diameter of 2cm plus 1cm of penumbra. The size of the spot is sufficient to uniformly irradiate the whole FPGA chip out of the beam. This is essential for preventing neutron-induced errors on power switches which may compromise the experiment. Moreover, having the DDR memory out of the beam allowed us to use it as a safe temporary storage of the Microblaze instruction code, thus preventing common failures induced by errors affecting the original instruction code. The workload of the system has been settled with an FFT application cycle executing the following operations: data memory initialization, data memory checking, two execution of Fast Fourier Transform (FFT) using Microblaze resources, checking differences between FFT applications, FFT execution using multipliers, result comparison. After the FFT application a set of configuration memory operations ant test routines are executed.

In details, configuration memory operations consist of three steps: N configuration memory frames read and on a signature computed, N configuration memory frames are written and signature stored, finally the same frames are read back again and a signature is compared. Signatures comparison allows detecting SEFI affecting the internal configuration access ports. Finally, the test routines include Software-Based Self-Test (SBST) programs suitable written for the DDR memory controller, Multipliers and ICAP port in order to individuate errors that cannot be covered by the FFT execution. The total execution time of this workload lasts for 10 seconds, during the radiation beam, the workload is continuously repeated until it stops due to a critical SEFI or a configuration memory scrub cycle.

Table II. Experimental Results – Long Duration Test

DRPM System – Average Maximal Duration [min]			
Xilinx Virtex-4 FX100		Xilinx Virtex-5 LX50T	
Original	Hardened	Original	Hardened
35.36	67.33	34.33	66.83

The analysis we performed consists on two experimental campaigns. The first aims at evaluating the resilience of the DRPM under the beam before the system incurs in a permanent error stopping the DRPM functionality. For this purpose the application is executed continuously without FPGA's configuration memory scrubbing. Results related to this analysis are reported in Table I, the hardened DRPM system results two times more resilient than the original ones with respect to critical SEFIs.

Table III. Experimental Results – Periodic Scrubbing every 20 minutes

	Xilinx Virtex-4 FX100		**Xilinx Virtex-5 LX50T**	
Average Time [mins]	**Original**	**Hardened**	**Original**	**Hardened**
Transient Error	10.07	9.68	9.87	9.43
Permanent Error	13.81	15.85	13.63	15.26
SEFI (Critical Error)	11.31	19.21	11.02	19.18
Errors [#]	**Original**	**Hardened**	**Original**	**Hardened**
Transient Error	89	93	93	96
Permanent Error	78	75	83	76
SEFI (Critical Error)	16	4	18	5

The second experiment is oriented to evaluate the overall performance of the system given a defined scrub cycle of 20 minutes, corresponding to 120 consecutive workload executions. Experimental results of the second radiation campaigns are reported in Table III, where we indicated the average time and the number of errors. Errors are classified according following classification: transient, in case it temporary affects the application execution; permanent, if it affects the execution until a scrub cycle is performed but it is detected by the checking operations or by the testing routines; SEFI in case it stops the execution of the DRPM system in an irreversible way (i.e. the processor hangs and it is necessary to reconfigure the entire DRPM). As illustrated in Table III, the hardened DRPM results about 4 times more resilient than the original version if critical SEFIs induced by FPGA's configuration memory bitflips are considered. The first part of the Table III includes the average time measured before one kind of error is observed into the system. The two versions have a similar number of transient errors since the system has not been protected versus these effects, besides these errors are detected by the checking operations executed during the Microblaze workload and by the testing routines. Furthermore,

it is interesting to notice that the ICAP port has an extremely low sensitive rate. It must be underline that the relevant comparison is related to the SEFI inducing critical error, considering the number of errors generated (Table III, last line). According to the achieved results, the hardened circuit is 3.8 times better than the original ones since only 5 versus 18 configuration memory upsets lead to the stopping of the functionality.

As reported in Table III, the number of transient errors result larger in the hardened version than the original ones. The controversy data must be analyzed considering that the hardened version has Microblaze with duplicated data path and the number of storage resources of the hardened version results bigger than the plain one. Please note that the number of transient errors result greater than the number of critical configuration memory bits, since the system has been hardened with respect to configuration memory upsets, vice versa, the system has not been hardened versus transient errors, since Flip-Flops and local processor memory have not been hardened with TMR or other redundancy based techniques.

VIII. Conclusions and Future Works

The adoption of reconfigurable systems on space and avionic applications is able to provide a great benefit thanks to the on-site reconfigurable capabilities. Reconfigurable systems are effectively implemented using Dynamic Reconfigurable Platforms or Very Long Instruction Word (VLIW) processors and requires appropriate validation in order to estimate radiation sensitivity in particular focusing on soft-errors and latch-ups. On the other side, these systems require the adoption of diagnostic capability, in order to provide the possibility to reconfigure the system and repair faulty units.

References

[1] D. Sabena, M. Sonza Reorda, L. Sterpone, "On the development of diagnostic test programs for VLIW processors, " Very Large Scale Integration (VLSI-SoC), 2013 IFIP/IEEE 21st International Conference on, pp. 84 – 89, 2013.

[2] M. Sonza Reorda, L. Sterpone, A.Ullah, "An error-detection and self-repairing method for dynamically and partially reconfigurable systems," Test Symposium (ETS), 2013 18th IEEE European, pp. 1 – 7, 2013.

[3] D. Sabena, M. Sonza Reorda, L. Sterpone, "On the Automatic Generation of Optimized Software-Based Self-Test Programs for VLIW Processors," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 22 , no. 4, pp. 813 – 823, April 2014.

[4] V. Stamatis, S. Wong, G. Gaydadjiev, K. Bertels, G. Kuzmanov and Elena Moscu Panainte, "The MOLEN Polymorphic Processor", IEEE Transactions on Computers, 53, pp. 1363 – 1375, 2004.

[5] S. Wong, T. Van As, G. Brown, "r-VEX: A Reconfigurable and Extensible Softcore VLIW Processor", ICECE Technology, 2008, FPT, International Conference on Field Programmable Technoogy, pp. 369 – 372, December 2008.

[6] S. Vassiliadis, S. Wong, S. Cotofana, "Microcode Processing: Positioning and Directions", IEEE Micro, pp. 21 – 30, July, 2003.

[7] F. Anjam, S. Wong, L. Carro, G.L. Nazar and M. B. Rutzig, "Simultaneous Reconfiguration of Issue-Width and Instruction Cache for VLIW Processor", Embedded Computer Systems (SAMOS), 2012, International Conference on, pp. 183 – 192, July 2012.

[8] Anthony Brandon and Stephan Wong, "Support for Dynamic Issue Width in VLIW Processors using Generic Binaries", IEEE Design, Automation and Test in Europe, 2013, pp. 827 – 832, March 2013.

[9] Fakhar Anjam and Stephan Wong, "Configurable Fault-Tolerance for a Configurable VLIW Processor", in Philip Brisk, Josè Gabriel Figueiredo Coutinho and Pedro C. Diniz, "Reconfigurable Computing: Architectures, Tools and Applications", Vol. 7806, Lecture Notes in Computer Science, pp. 167 – 178, Springer Berlin, Heidelberg, 2013.

[10] Jiri Gaisler, Sandi Habic and E. Catovic, "GRLIB IP Library User's Manual", http://gaisler.com/products/grlib/grlib.pdf, 2010.

[11] A. Avizienis, J.-C. Laprie, B. Randell and C. Landwehr, "Basic concepts and taxonomi of dependable and secure computing", IEEE Transactions on Dependable and Secure Computing, Vol. 1, No. 1, pp. 11 – 33, January 2004.

[12] M. Glab, "Dependability-aware system-level design for embedded systems", Dissertation, University of Erlangen-Nuremberg, Germany, Mar. 2011, Verlag Dr. Hut, Munich, Germany.

[13] R. Glein, F. Rittner and A. Hoffmann, "Ensuring FPGA reconfiguration in space", Xcell Journal, No. 84, pp. 23 – 27, July 2013, [online] Available: http://www.xilinx.com/pubblications/archives/xcell/Xcell84.pdf

[14] G. Swift and G. Allen, "Virtex-5QV Static SEU characterization summary", May 2013, [online], Available: http://parts.jpl.nasa.gov/wp-content/uploads/V5QV-Static-SEU-Summary-ReportRevD.pdf

[15] R. Glein, B. Schmidt, F. Rittner, J. Teich, and D. Ziener, "A self-adaptive SEU mitigation system for FPGAs with an internal block RAM radiation particle sensors", in 2014, IEEE 22st Annual International Symposium on Field-Programmable Custom Machines (FCCM), Boston, May, 2014.

[16] M. Koester, W. Luk, J. Hagemeyer, M. Porrmann, U. Ruckert, "Design Optimizations for Tiled Partially Reconfigurable Systems", IEEE Transactions on VLSI, 2010, pp. 1 – 14.

[17] G. R. Allen, G. Madias, E. Miller, G. Swift, "Recent Single Event Effects Results in Advanced Reconfigurable Field Programmable Gate Arrays", IEEE Radiation Effects Data Workshop, 2011, pp. 1 – 6.

[18] M. Berg, C. Poivey, D. Petrick, D. Espinosa, A. Lesea, K. A. LaBel, M. Friendlich, H. Kim, A. Phan, "Effectiveness of Internal Versus External SEU Scrubbing Mitigation Strategies in a Xilinx FPGA: Design, Test and Analysis", IEEE Transactions on Nuclear Science, Vol. 55, Issue 4, Part 1, 2008, pp. 2259 – 2266.

[19] H. Quinn, K. Morgan, P. Graham, J. Krone, M. Caffrey, K. Lundgreen, "Domain Crossing Errors: Limitations on Single Device Triple-Modular Redundancy Circuits in Xilinx FPGAs", IEEE Trans. on Nuclear Science, Vol. 54, Issue 6, Part 1, 2007, pp. 2037 – 2043.

[20] Y. Cao, P. Bose and J. Tschanz, "Reliability Changes in Nano-CMOS Design (Guest Editor's Introduction), IEEE Design & Test of Computers, 26 (6), pp. 6-7, 2009.

[21] I. Koren and C. M. Krishna, "Fault-Tolerant Systems", Morgan Kaufmann, 2007.

[22] L. Guerra, M. Potkonjak and J. M. Rabaey, "High Level Synthesis Technique for Reconfigurable Datapath Structures", IEEE Conference on Computer Aided Design (ICCAD'93), pp. 26-29, 1993

[23] L. Guerra, M. Potkonjak and J. M. Rabaey, "Behavioral-Level Synthesis of Heterogeneous BISR reconfigurable ASIC's", IEEE Transactions on Very Large Scale of Integration (VLSI), 6(1), pp. 158 – 167, 1998.

[24] A. Meixner and D. J. Sorin, "Detouring: Traslating Software to Circumvent Hard Faults in Simple Cores", Proceedings of the International Conference on Dependable Systems and Networks (DSN), pp. 80 – 89, 2008.

[25] S. Muller, M. Scholzel and H. T. Vierhaus, "Towards a Graceful Degradable Multicore-System by Hierarchical Handling of Hard Errors", Proceedings of 21st Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP'13), 2013.

[26] P. Bernardi, E. Sanchez, M. Schillaci, et. al.,"An effective technique for minimizing the cost of processor software-based diagnosis in SoCs", Proceedings of the Conference on Design, Automation and Test in Europe (DATE'06), pp. 412 – 417, 2006.